P1 *Escolha múltipla (4 valores)*

NOTA IMPORTANTE: em caso de dúvida escolha a opção que lhe parecer mais completa.

1.a) A documentação completa do projecto de um programa é composta por: *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☒
☐
☐

1.b) Se um microprocessador só entende linguagem máquina, como é que executa programas escritos em linguagens de alto nível como por exemplo a do Matlab, C++, Fortran, etc.? *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☒
☐
☐

1.c) Um erro de execução num programa ocorre normalmente em que situação? *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☐
☒
☐

1.d) A lista de tarefas que é utilizada em conjunto com a sequenciação, a selecção e a repetição para construir o algoritmo é obtida: *(0.5v)*
Seleccione a resposta correcta: ☒

☒
☐
☐ I
☐

1.e) Quando escolheria utilizar uma estrutura de selecção `if` em vez de uma `switch`? *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☐
☒
☐
☐

1.f) A operação de abertura de um ficheiro serve para: *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☐
☐
☒
☐

1.g) No algoritmo de ordenação por indexação: *(0.5v)*
Seleccione a resposta correcta: ☒

☒
☐
☐
☐
☐

1.h) Uma *function handle* pode ser utilizada para passar funções como argumentos para outras funções em Matlab. Que tipos de funções podem ser passadas desta forma? *(0.5v)*
Seleccione a resposta correcta: ☒

☐
☐
☐
☒
☐

P2 *Análise e depuração de código (5 valores)*

2.a) Apresente o(s) resultado(s) enviado(s) para o monitor em cada um dos seguintes casos.

```
a = ['a' 'b' 'c'; 'd' 'e' 'f'];
fprintf('%s %c',a(2,:),a(1,3));
```

RESPOSTA: *(0.5v)*

```
def c
```

```
a = [1 2 3; 4 5 6; 7 8 9];
b = zeros(size(a));
for i = 1:3
    for j = 1:3
        if j >= i
            b(i,j) = a(i,j);
        end
    end
end
disp(b)
```

RESPOSTA: *(0.5v)*

```
1   2   3
0   5   6
0   0   9
```

```
a = [8 3 1];
for i = 2:length(a)
    b(i) = a(i)-a(i-1);
end
disp(b)

for i = length(a)-1:-1:1
    c(i) = a(i+1)-a(i);
end
disp(c)
```

RESPOSTA: *(0.5v)*

```
 0     -5     -2

-5     -2
```

```
a = 3;
b = 2;
c = @(a,b) a^2+b^2;

[e f] = func1(a,b,c);
disp(e)
disp(f)
```

RESPOSTA: *(0.5v)*

29

7

func1.m

```
function [w z] = func1(x,y,z)
   x = x + y;
   w = z(x,y);
   z = x + y;
```

```
a(1) = struct('os','andrio','ide',[]);
a(2).os = 'janelas';
a(2).ide = {'nm',[1 3 2],'prc',543.0};
a(1).ide = {[3 4 1],'nm'};

disp(upper(a(2).os(3:2:end)))
disp(a(1).ide{1}(2)*2)
```

RESPOSTA: *(0.5v)*

NLS
8

```
func2('algebra')
```

RESPOSTA: *(0.5v)*

func2.m

```
function func2(plv)

if length(plv) == 1
    fprintf('%c',plv);
else
    fprintf('%c',plv(end));
    func2(plv(1:end-2));
end
```

abga

2.b) Cada um dos programas seguintes tem **cinco erros**. Assinale e corrija os erros em cada caso para que os programas executem correctamente.

NOTA IMPORTANTE - Para modificar o código deve proceder do seguinte modo:

- **Na sua resposta deve apenas indicar o número da linha se o código estiver correcto, caso o código esteja incorrecto deve indicar a linha e a respectiva alteração.**
- **Caso necessite de adicionar novas linhas numere-as utilizando a linha anterior como base, seguida de um ponto e das letras a, b, etc.**

```
1: letra = input('Resposta(S/N): ',s);
2: if letra == 's' || letra == 'S'
3:     disp('Continuar')
4: elseif letra == 'n' | letra == 'N'
5:     disp('Parar')
6: else
7:     disp(Erro)
8: end
```

RESPOSTA

```
1: letra = input('Resposta(S/N): ','s');
2:
3:
4: elseif letra == 'n' || letra == 'N'
5:
6:
7:    disp('Erro')
8:
```

RESPOSTA: *(1.0v)*

```
1:  % Cria um vector de estruturas com
2:  % coordenadas de pontos. Quando termina
3:  % guarda-o num ficheiro do Matlab.
4:
5:  aux = [];
6:  while true
7:    fprintf('1 - Novo ponto\n');
8:    fprintf('2 - Terminar\n');
9:    opcao = input("Introduza a opcao:');
10:
11:   switch opcao
12:     case {1}
13:       fprintf('Introduza:\n');
14:       aux.x = input('x: ');
15:       aux.y = input('y: ');
16:       pontos = [pontos aux.x];
17:
18:     case: 2
19:       save pontos;
20:
21:     end
22: end
```

```
1:
2:
3:
4:
5: pontos = [];
6:
7:
8:
9: opcao = input('Introduza a opcao:');
10:
11:
12:
13:
14:
15:
16: pontos = [pontos aux];
17:
18: case 2
19:
20: break
21:
22:
```

RESPOSTA: *(1.0v)*

```
1: function c = leCabecalho(fich)
2: % Lê os dados do cabeçalho dos
3: % ficheiros .tab do projecto para
4: % um cell array de strings c
5:
6: c = {};
7: id = fopen(fich,'wt');
8: if id == -1
9:   fprintf('Não foi aberto!');
10: else
11:   while true
12:     lin = fgetl(id);
13:     lin = strtrim(fich);
14:     if strncmp(lin,...
15:               'OBSERVATION_NAME',16)
16:       c{1} = lin;
17:     elseif strncmp(lin,...
18:                     'IMAGE_TIME',10)
19:       c{2} = lin(2);
20:     elseif strncmp('END',lin,1)
21:       exit;
22:     end
23:   end
24:   n_fechou = fclose(id);
25:   if n_fechou
26:     fprintf('Não fechou!');
27:   end
28: end
```

```
7: id = fopen(fich);
13: lin = strtrim(lin);
19: c{2} = lin;
20: elseif strncmp('END',lin,3)
21: break;
```

NOTA1: `strncmp(S1,S2,N)` devolve valor verdadeiro se os primeiros N caracteres de S1 e S2 forem iguais.

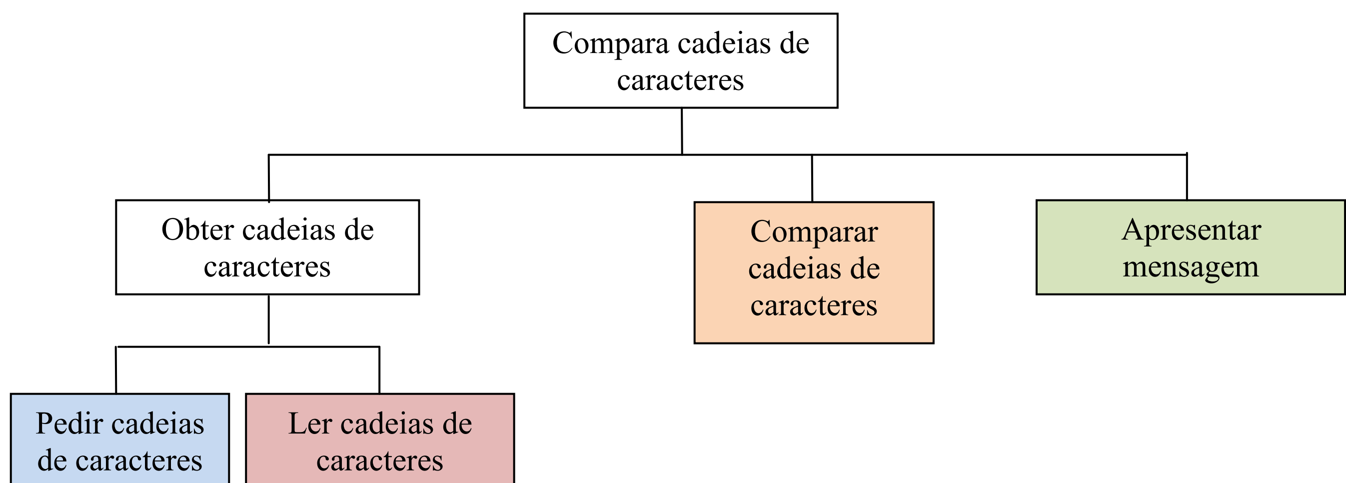
NOTA2: `strtrim(S)` remove os espaços em branco insignificantes de S.

P3 *Análise de Problema e Algoritmo (3,5 valores)*

NOTA IMPORTANTE: a robustez dos programas ao tipo de entrada apenas deve ser considerada quando explicitamente pedida no enunciado da pergunta!

3. Crie o algoritmo para um programa que peça ao utilizador duas cadeias de caracteres quaisquer, e que apresente uma mensagem indicando se as cadeias de caracteres são iguais ou diferentes. Assuma que não é possível comparar directamente as cadeias de caracteres, e que a comparação tem que ser feita letra-a-letra. O programa só deve comparar as cadeias de caracteres se tiverem o mesmo comprimento (assuma que é possível saber o comprimento), e deve parar a comparação e apresentar a mensagem logo que descubra se as cadeias de caracteres são iguais ou diferentes.

3.a) Abordagem Top-down *(0.8v)*



3.b) Esquema de processamento *(0.5v)*

Entradas: **cadeia1, cadeia 2 – valores pedidos ao utilizador.**

Saídas: **Mensagem indicando se as cadeias são iguais ou diferentes.**

3.c) *(2.2v)*

Algoritmo do corpo principal do programa:

1) Pedir ao utilizador valores para **cadeia1,** e **cadeia2**
2) Receber valores para **cadeia1,** e para **cadeia2**
3) Se o comprimento de **cadeia1** for diferente do de **cadeia2**
    a. Apresentar mensagem indicando que as cadeias são diferentes

   Caso contrário
    a. Inicializar **iguais** = verdadeiro e **k** = 1
    b. Repetir enquanto **iguais** for verdadeiro e **k** <= comprimento da **cadeia1**
        i. Se **cadeia1**(k) ≠ **cadeia2**(k)
            1. **iguais** = falso
        ii. **k** = **k** + 1
    c. Se **iguais** for verdadeiro
        i. Apresentar mensagem indicando que as cadeias são iguais
    d. Caso contrário
        i. Apresentar mensagem indicando que as cadeias são diferentes

P4 *Programação (6,5 valores)*

NOTA IMPORTANTE: a robustez dos programas ao tipo de entrada apenas deve ser considerada quando explicitamente pedida no enunciado da pergunta!

4.a) Para um prisma que tenha como base um polígono com n lados e altura h, o volume V e a área da superfície A são dados respectivamente por:

$$V = \frac{n}{4} h S^2 \cot\frac{\pi}{n}$$
$$A = \frac{n}{2} S^2 \cot\frac{\pi}{n} + n\,S\,h$$

onde S é o comprimento dos lados do polígono. Escreva uma **<u>função</u>** chamada **`prismaVolArea`**, que receba como **<u>argumentos de entrada</u>** valores para n, h e S, e que devolva como **<u>argumentos de saída</u>** os valores do volume V e da área A **<u>por esta ordem</u>**. Assuma que todas as unidades são coerentes.

(1.8v)

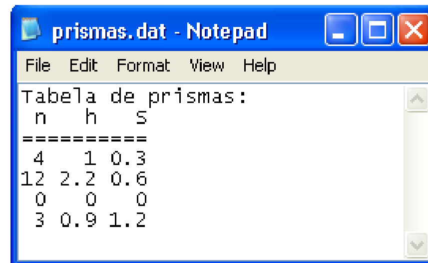
NOTA: a função que devolve o valor da co-tangente para um argumento em radianos no Matlab é denominada **`cot`**.

```
function [V, A] = prismaVolArea(n, h, S)

V = n/4*h*S.^2*cot(pi/n);

A = n/2*S.^2*cot(pi/n)+ n*S*h;
```

4.b) Escreva uma **<u>função</u>** chamada **`leDados`** que **<u>leia de um ficheiro</u>** chamado `prismas.dat` valores para um vector de estruturas. Cada elemento do vector de estruturas deve possuir os campos **`n`**, **`h`** e **`S`**, onde se guardarão os valores **<u>numéricos</u>** respectivos lidos de uma linha de dados do ficheiro. Existem no ficheiro linhas de dados com valores não válidos todos iguais a 0, estes dados não devem aparecer no vector de saída. O ficheiro é **<u>formatado</u>** conforme o exemplo seguinte, mas o número de linhas de dados pode ser qualquer. Apenas o cabeçalho terá sempre o número de linhas do exemplo.



A função deve possuir um único **<u>argumento de saída</u>** que é todo o vector de estruturas. Terá que abrir e fechar o ficheiro de dados, naturalmente, mas não necessita verificar ou dar mensagens sobre as operações de abertura e de fecho.

Pode utilizar qualquer função sua conhecida para ler os dados do ficheiro, desde que a saída da função seja o vector de estruturas pedido.

(2.2v)

```
function prisma = leDados

fid = fopen('prismas.dat');

for i = 1:3
    linha = fgetl(fid);
end

k = 1;
while ~feof(fid)
    linha = fgetl(fid);
    prisma(k).n = str2num(linha(1:2));
    prisma(k).h = str2num(linha(3:6));
    prisma(k).S = str2num(linha(7:end));

    if prisma(k).n ~= 0 && prisma(k).h ~= 0 && prisma(k).S ~= 0
        k = k + 1;
    end
end

fclose(fid);
```

4.c) Escreva um ***script*** que leia do ficheiro `prismas.dat` um vector de estruturas contendo informação sobre prismas **utilizando** a função **`leDados`** definida na alínea **4.b)**.

Em seguida o programa deve indicar ao utilizador quantos prismas há no vector (não é preciso mostrá-los, basta dizer quantos há), e pedir-lhe o índice do prisma que pretende avaliar. O programa deve então calcular e apresentar ao utilizador com uma precisão de 3 casas decimais o valor da área do prisma seleccionado, calculada **utilizando** a função **`prismaVolArea`** definida na alínea **4.a)**.

O programa deve em seguida definir um vector cujos valores variem desde 70% até 130% de *S* do prisma seleccionado, em passos de 10% de *S*, e apresentar num gráfico a correspondente curva de volume em função de *S,* sendo também estes volumes calculados **utilizando** a função **`prismaVolArea`** definida na alínea **4.a)**. O gráfico deve ter um título adequado, assim como rótulos para os eixos horizontal e vertical.

Finalmente o programa deve perguntar ao utilizador se deseja terminar, dando a hipótese de responder (S/N) em maiúsculas ou minúsculas. Caso o utilizador responda que não o programa volta ao início, pedindo novamente o índice do prisma que se pretende avaliar.

Não necessita verificar as opções do utilizador, nem na escolha do prisma nem na saída.

Note que a **utilização** das funções só requer o conhecimento dos respectivos **nomes, e argumentos de entrada e saída** que podem ser consultados nos enunciados das alíneas anteriores. **Não defina qualquer função na resposta a esta alínea**.

(2.5v)

```
prisma = leDados;

terminar = false;
while ~terminar
    fprintf('Há %d prismas no vector.\n',length(prisma));
    i = input('Qual o prisma que deseja analisar? ');
    n = prisma(i).n;
    h = prisma(i).h;
    S = prisma(i).S;

    [V, A] = prismaVolArea(n, h, S);
    fprintf('A = %.3f\n', A);

    S_vec = [0.7:0.1:1.3]*S;
    [V_vec A_vec] = prismaVolArea(n, h, S_vec);
    plot(S_vec, V_vec);
    title('Volume do prisma em função do lado da base');
    xlabel('Comprimento do lado da base');
    ylabel('Volume do prisma');

    opcao = input('Deseja terminar? (S/N) ','s');
    if opcao ~= 'N' && opcao ~= 'n'
        terminar = true;
    end
end
```

P5 *Programação de GUI (1 valor)*

5. Explique resumidamente o que entende por programação dirigida por eventos (*event-driven*), e qual a principal diferença deste tipo de programação para a programação de execução sequencial.

Programação *event-driven*

- Os eventos incluem acções do utilizador, acções devidas a outros elementos de código, ou acções externas (i.e. geradas pelo SO), como por exemplo a criação de um ficheiro ou a ligação de *hardware* periférico.
- O controlo do programa não depende de um fluxo de execução programado no código mas sim da sequência de elementos de código (*callbacks*) disparada por reacção aos eventos.
- O controlo do fluxo da execução do programa constitui a principal diferença entre a **programação dirigida por eventos** e a **programação de execução sequencial**.

TÉCNICO LISBOA Computação e Programação 2012 / 2013 8